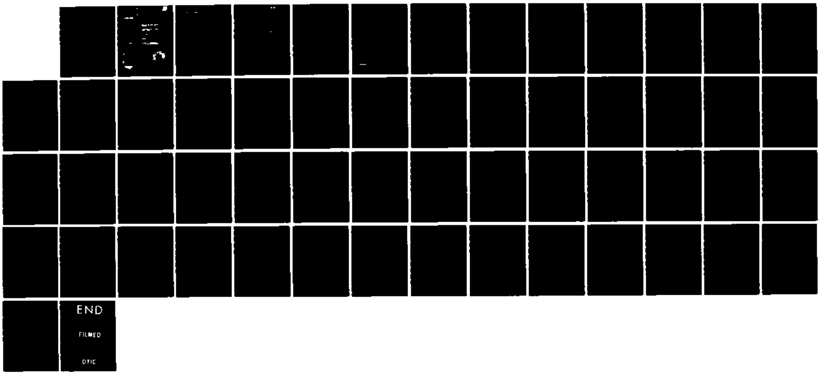
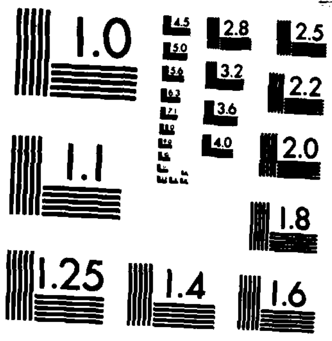

END
FILMED
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

[illegible]

Published by The Rand Corporation

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER N-2099-NA	2. GOVT ACCESSION NO. AD-A153088
	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Overview Of RSAC System Software: A Briefing	5. TYPE OF REPORT & PERIOD COVERED Interim
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) H. Edward Hall, Norman Z. Shapiro, Herbert J. Shukiar	8. CONTRACT OR GRANT NUMBER(s) DNA001-80-C-0298
9. PERFORMING ORGANIZATION NAME AND ADDRESS The Rand Corporation 1700 Main Street Santa Monica, CA. 90406	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Director of Net Assessment Office of the Secretary of Defense Washington, D.C. 20301	12. REPORT DATE January 1985
	13. NUMBER OF PAGES 45
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release: Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

No Restrictions

18. SUPPLEMENTARY NOTES

[illegible]

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Programs,	Strategic Analyses
Computerized Simulation,	Artificial Intelligence,
War Games.	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See Reverse Side

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

This Note gives an overview of the Rand Strategy Assessment Center's (RSAC) software design and summarizes characteristics of the automated war gaming system. It reviews basic concepts and requirements and examines three different aspects of the RSAC software design: (1) System interface, including the complex communication among the RSAC system's various software modules; (2) the communication mechanism which promotes modularity, thus simplifying system integration, evolution, and maintenance; (3) system control, which treats game phenomena with extensive use of co-routines managed by a coordinating program. The Note discusses interactive gaming with human teams, identifying several classes of interaction and describing techniques that promote it. Finally, it discusses the overall computing milieu characterized by operations on VAX 11-750 or VAX 11-780 computers, the C programming language, Unix operating system, a computer code requiring about 2 megabytes of random access memory, and the potential for eventual operations on geographically separated microprocessors. Originator-supplied key words included. → front

A RAND NOTE

OVERVIEW OF RSAC SYSTEM SOFTWARE: A BRIEFING

H. Edward Hall, Norman Z. Shapiro,
Herbert J. Shukiar

January 1985

N-2099-NA

Prepared for The Director of Net Assessment,
Office of the Secretary of Defense

Rand
1700 MAIN STREET
P.O. BOX 2138
SANTA MONICA, CA 90406-2138

PREFACE

This Note documents a briefing presented on October 19, 1983, to the Department of Defense Working Group overseeing progress of the Rand Strategy Assessment Center (RSAC). The presentation was an attempt to provide a high-level picture of the RSAC system software. More technically detailed descriptions will be published elsewhere.

The RSAC itself is an ambitious, high-risk, multiyear effort to develop improved methods for strategic analysis--taken in its broadest sense--by combining the best features of war gaming and analytic modeling. The centerpiece of the effort is an emerging capability for *automated war gaming* in which, depending on the application, some or even all of the human teams can be replaced by artificial intelligence models, thereby allowing systematic and reproducible analysis.

The RSAC effort is supported by the Director of Net Assessment, in the Office of the Secretary of Defense, under Contract No. DNA001-80-C-0298. Comments on this Note are welcome; they should be directed to the authors or to Dr. Paul K. Davis, Director of the Rand Strategy Assessment Center.

SUMMARY

Rand's Strategy Assessment Center (RSAC) is developing an operational prototype system for automated war gaming under a multiyear DoD contract. A major milestone in that work is the completion of an overall software design consistent with the ambitious RSAC concepts developed earlier.[1] This briefing book gives an overview of the RSAC's software design and summarizes characteristics of the automated war gaming system. After a short review of basic concepts and requirements, it examines the RSAC software design from three different perspectives.

The first perspective deals with system interfaces, describing the complex kinds of communication among the various RSAC system's software modules. Those modules include three rule-based agents that represent *Red, Blue*, and *third-country* behavior. The rule-based agents select courses of action (plans) based on the world situation and can adjust plans in response to changes in the world situation. A fourth module simulates the interaction of the rule-based agents' plans.

The second perspective addresses the mechanism the system employs to permit communication among the agents. This mechanism requires the modules to communicate indirectly via a database called the *World Situation Dataset*. No direct communication among the agents is permitted. This database approach promotes modularity, simplifying system integration, evolution, and maintenance. The approach can also facilitate distributed processing where one or more agents reside on different computers.

The third perspective examines the issue of system control, something of considerable interest to computer scientists. The RSAC system treats game phenomena with extensive use of co-routines managed by a coordinating program called *System Monitor*. The design allows the nations in the game to do projections or "look-aheads" employing a game within a game and models of each superpower's perceptions regarding the

[1]See Paul K. Davis and James A. Winnefeld, *The Rand Strategy Assessment Center: An Overview and Interim Conclusions about Utility and Development Options*, The Rand Corporation, R-2945-DNA, March 1983.

ACKNOWLEDGMENTS

Many Rand colleagues have contributed to the design and implementation of the RSAC system software; it would be impossible to acknowledge them all. However, the following deserve special mention:

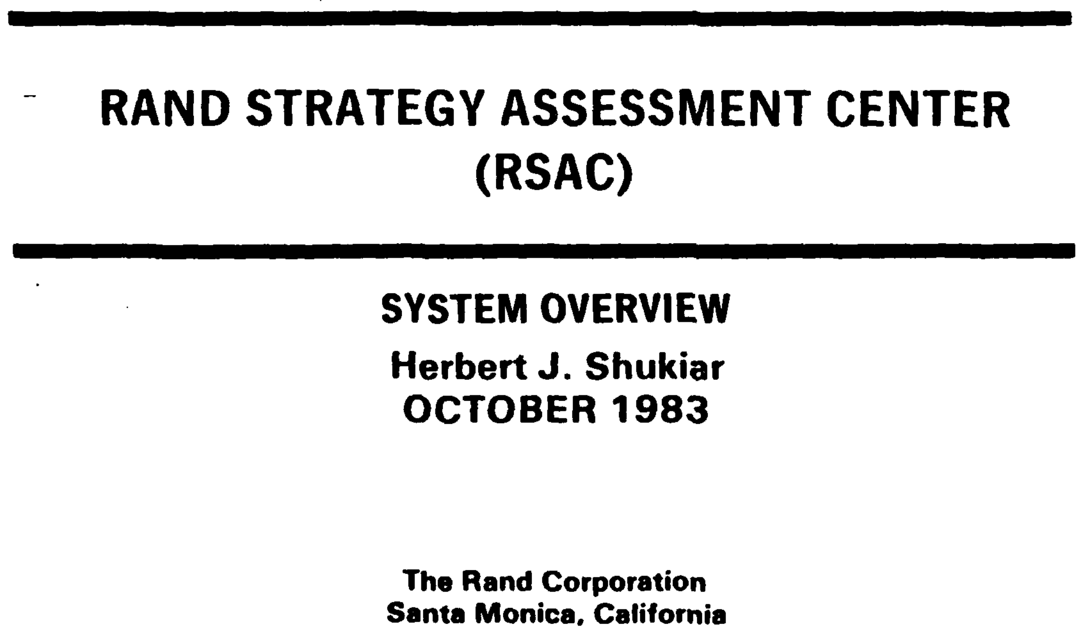
Arthur M. Bullock
Jean L. LaCasse
Mark LaCasse
Christe A. McMenomy

The design has also enjoyed the influence of those whose ideas and participation have played a major role in shaping the system architecture:

Steven C. Bankes	William L. Schwabe
Bruce W. Bennett	David A. Shlapak
Paul K. Davis	Randall Steeb
James Gillogly	James A. Winnefeld
William M. Jones	

CONTENTS

SLIDE 1: SYSTEM OVERVIEW



RAND STRATEGY ASSESSMENT CENTER (RSAC)

SYSTEM OVERVIEW

Herbert J. Shukiar
OCTOBER 1983

The Rand Corporation
Santa Monica, California

This briefing book presents an overview of system software in the Rand Strategy Assessment Center (RSAC).

SLIDE 2: BRIEFING OUTLINE

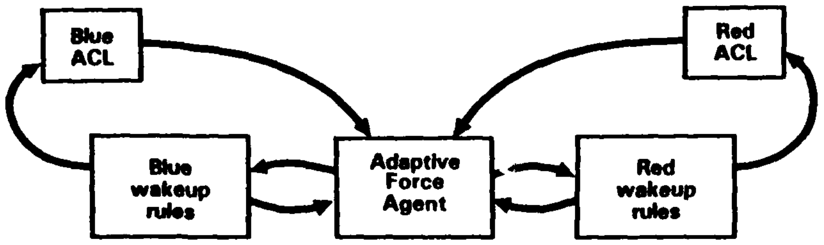
- **Brief review**
- **RSAC system interfaces**
- **Managing interface complexity**
- **System control**
- **Interactive games**

First, we briefly review some RSAC concepts that were described during an earlier briefing to the sponsor and the oversight Working Group. Next, we describe the RSAC system from three different perspectives, indicated by the second through fourth bullets on the slide. To gain a thorough understanding of the RSAC system, all three perspectives must be understood. Finally, we examine and summarize the area of interactive games and how the RSAC system will permit the conduct of such games.

convenient for the Force Agent to manage the exercise clock, i.e., incrementing exercise time.

The Red and Blue wakeup rules permit the two major agents to monitor the unfolding situation. These rules, written in ABEL, examine those aspects of the unfolding situation that the rule-writer deemed important. At each *tick* of the exercise clock, the Force Agent tests both the Red and Blue wakeup rules to determine if the major agents wish to be awakened. The double arrows between the Force Agent and each major agent are there to reflect Force Agent's continual testing of these wakeup rules. If a major agent wishes to be awakened, then Force Agent will relinquish control, thereby permitting that agent to look at the current world situation and adapt appropriately. A major agent will not be awakened unless its wakeup rules *fire*.

SLIDE 11: RSAC SYSTEM INTERFACES MAJOR AGENTS TO FORCE WITH MAJOR AGENT WAKEUP RULES



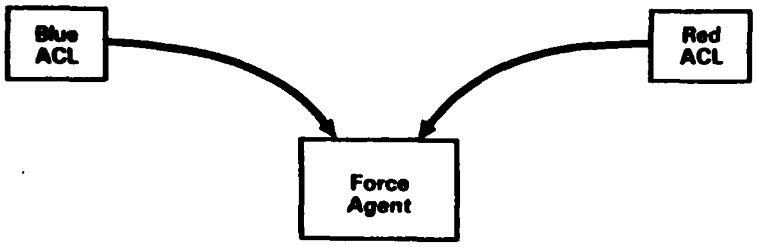
This slide illustrates how the major agents and Force Agent interact. It shows the Blue and Red wakeup rules monitoring the unfolding world situation and also notes that the RSAC Force Agent is *adaptive* in that it permits the major agents to alter their courses of action in response to the unfolding conflict.[1]

On slide 10 the ACLs passed complete operations plans to the Force Agent. On this slide, however, the ACLs pass appropriate *segments* of plans, segments determined by Red and Blue rule-writers who wish to schedule opportunities for adapting to changing circumstances. Segments correspond to real-world phases such as alert, deployment, initial execution, and termination.

Force Agent reacts to instructions calling for changes in Red and Blue assets. It alters the status of these military assets by evaluating time dependent difference equations, algorithms representing the movement of the military assets from the current status to the desired status. Hence we can view the process as *goal-directed simulation*. Because the difference equations are time dependent, it is

[1]One might argue that the major agents are adaptive rather than the Force Agent, since it is the major agents that adapt to the unfolding world situation. This is certainly a reasonable perspective, but we have chosen to call the Force Agent *adaptive* because it permits such monitoring and adjustment.

SLIDE 10: RSAC SYSTEM INTERFACES MAJOR AGENTS TO FORCE



Slides 10-13 present a picture of how the various agents interact. Slide 10 shows a simplified Force Agent accepting orders from the Blue and Red ACLs. In this slide the ACLs pass complete operations plans to the Force Agent, which simulates the interactions of those plans and does not permit the major agents to monitor or adjust to the unfolding world situation, i.e., Force Agent acts as a *black box* model with no major agent interaction. This slide serves as a reference point upon which the next three slides build.

SLIDE 9: BRIEFING OUTLINE

- **Brief review**
- ✓ **RSAC system interfaces**
- **Managing interface complexity**
- **System control**
- **Interactive games**

This completes the review. We next examine the RSAC software system from three perspectives, the first of which deals with RSAC system interfaces.

nuclear exchange, the Force Agent may reduce this time step to five minutes.

The Force Agent is, by design, a highly aggregated model of Force status and interactions. A highly detailed model not only requires substantial computer capacity and running time but also is inappropriate for most strategic analyses.

SLIDE 8: FORCE AGENT

- **Simulates precombat and combat activities**
- **Responds to orders from Red, Blue, and Scenario Agents—does not act independently**
- **Controls clock**
  - **Variable time step based on activity being simulated**
  - **Checks for Major Agent wakeup at each clock tick**

The Force Agent simulates the precombat and combat activities of the Red, Blue, and Scenario Agents, responding to orders from those agents and permitting those agents to monitor and adjust to the unfolding conflict situation. Once the rule-based agents have selected their courses of action, the Force Agent simulates the interactions of those courses of action.

As the Force Agent simulates these interactions, the three rule-based agents are said to be *asleep*, i.e., these agents are not executing. However, each rule-based agent has *wakeup rules*, ABEL rules that monitor the unfolding world state looking for situations that demand a rule-based agent's attention. The Force Agent invokes the wakeup rules at each tick of the exercise clock. When a wakeup rule fires, the Force Agent goes to sleep, thereby allowing the appropriate rule-based agent(s) to awaken and adjust the chosen course of action in response to the unfolding world state.

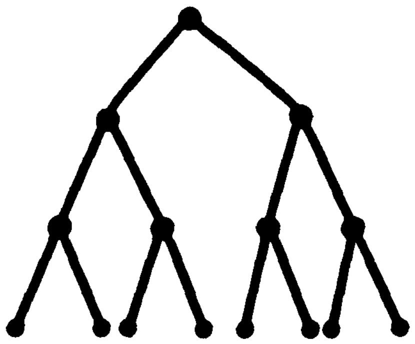
The Force Agent controls the exercise clock and employs a variable time step depending on the nature of the activity being simulated. For example, when the rule-based agents are not in an active combat state, the Force Agent may increment the clock in twelve-hour steps. During a

Blue to be ready for the attack. The Red plan can contain rules to test for these two discrete circumstances (Blue ready or not ready) and choose an appropriate course of action.

To illustrate continuous choices, suppose Red is in the midst of combat and must decide how to apportion his tactical air assets. Rather than have a large number of alternative discrete paths that are really modest variations of a basic air asset apportionment scheme, Red may simply choose to employ an algorithmic approach that looks at the current situation and apportions accordingly. The algorithm can use the various elements of the current situation as inputs to an algorithmic apportionment procedure, thereby eliminating the large number of discrete paths that would be needed were a discrete choice employed.

SLIDE 7: OPERATIONS PLANS

- **Direct military and political actions consistent with guidance, objectives, and strategy**
- **Can contain both "continuous" and discrete branch points**



The rule-based plans selected by the NCL for ACL implementation direct the agent's political and military actions. One can think of a plan first as a sequence of precise action instructions. However, a plan can also be characterized as a tree structure, with each node on the tree representing a decision the ACL must make--note that the NCL, in choosing plans, is working on a higher-level track. On the slide we have shown a plan whose nodes have only two alternative paths, but a node can have any number of such paths. Further, while the slide illustrates the situation where the choices are *discrete*, i.e., we must choose from two or more specified paths, the plan may also include *continuous* choices, i.e., choices that reflect a modest adjustment of parameters with no change of path.

To illustrate discrete choices, suppose Red is in the midst of mobilizing and wishes to determine whether to attack now or delay the attack until mobilization is completed. The Red Agent may choose to attack now if he believes he can catch Blue by surprise. On the other hand, Red may choose to wait for complete mobilization if he believes level for implementation. This plan is relatively simple and goal directed.

The middle level, or *Area Command Level* (ACL), implements the NCL-selected plan. The plan contains orders to the Force Agent as well as requests of Scenario Agent. While the high-level decisionmaking takes place at the NCL level, the ACL has choices as well. The next slide will review the operations plan briefly and describe the types of choices available to the ACL.

The lowest level is called the *Tactical Command Level* (TCL). This level is incorporated within the Force Agent. The Force Agent simulates the effects of the courses of action selected by the Red, Blue, and Scenario Agents. To do so, it contains models describing Force operations and interactions. Further, Force Agent manages the exercise clock. Therefore, the Force Agent manages the details of Force interactions as the conflict unfolds. This agent responds to orders issued by the Red, Blue, and Scenario Agents. Scenario Agent orders to the Force Agent can only give specified third-country military assets to one of the two major agents--we call the Red and Blue Agents *Major Agents* to reflect the RSAC's emphasis on superpower strategy assessment.

SLIDE 6: THREE-LEVEL SYSTEM STRUCTURE

- **National Command Level (NCL)**
  - **– Determines**
    - **Context (e.g., conflict in SWA)**
    - **Objectives (e.g., occupy FRG)**
    - **Strategy (e.g., standing start attack)**
  - **– Selects analytic war plan representing the chosen strategy to meet the specific objectives within the overall context**
- **Area Command Level (ACL)**
  - **– Implements the AWP selected by NCL**
  - **– Contains alternatives within it**
- **Tactical Command Level/Force Agent (TCL)**
  - **– Carries out detailed orders contained in AWP**

This slide discusses the approach employed in writing the Red and Blue rule sets. This approach divides the Red and Blue decisionmaking structure into three levels. Each agent therefore has a three-level decisionmaking hierarchy.

Each agent's top level, the *National Command Level* (NCL), accepts as *inputs* a current world situation and information about his own and his adversary's nature. These inputs permit the NCL to determine: (1) the context within which a decision must be made (for example, an ongoing conflict in Southwest Asia), (2) the operational objectives the agent wishes to achieve (for example, occupy FRG), and (3) the operational strategy the agent should employ to meet these objectives (for example, and greatly simplified, a standing start attack of the FRG). By determining the context, operational objectives and operational strategy, the NCL chooses an *analytic war plan* (AWP), which consists of rules for prosecuting the war. Once the NCL selects an AWP, a corresponding operations plan (or simply plan) is sent to the middle

SLIDE 5: RSAC AGENTS

- **Red Agent***
- **Blue Agent***
- **Scenario Agent***
- **Force Agent**

***Written in ABEL, a "C" compatible rule-based language**

This slide lists the various RSAC agents for completeness. The top three--*Red Agent, Blue Agent*, and *Scenario Agent*--capture Red, Blue, and third-country behaviors and are written in the ABEL rule-based language, developed specifically for RSAC needs. These can be thought of as *behavior* agents that reflect acts of volition on the part of the automated players. These agents determine the courses of action to follow. The Force Agent, on the other hand, does not reflect this volition. It rather simulates the combined effects of those courses of action.[1] The Force Agent is written in C.

[1]For a more general discussion of the modeling of *volition* as compared with the modeling of physical events, see Carl H. Builder, *Toward a Calculus of Scenarios*, The Rand Corporation, N-1855-DNA, January 1983, and in particular pp. 16-17.

SLIDE 4: RSAC SYSTEM REQUIREMENTS

- **Transportable**
  - **Single language**
  - **Widely available operating system**
- **Transparent and realistic**
  - **Easy to use man-machine interface**
  - **Rule-based Artificial Intelligence approach captures complex military and political behavior**
- **Modular**
  - **Separable agents with strict interface standards**
- **Modifiable**
  - **Strict adherence to structured programming standards**
  - **Strict data base access standards**
- **Automated and man-machine operation**
  - **System architecture compatible with both types of operation**

This slide lists the five system design requirements discussed at the March, 1983, Working Group meeting. We shall return to them at the end of this briefing and discuss our design's response to them.

To capture the behavioral richness of war gaming, we have taken a rule-based artificial intelligence approach. Subject-area experts write rules to capture Red, Blue, and third-country behavior. These rules, written in an Englishlike language called ABEL, provide a degree of transparency so that others not familiar with artificial intelligence can gain an intuitive understanding of the rule-writer's intent. The rule sets reflect expert understanding of how each adversary will respond to conflict. The three rule sets serve as the experts' *agents*, yielding automated representations of their expertise.

Integrated with these automated rule-based agents is an aggregated Force Model that simulates the detailed preconflict and conflict interactions of the three agents. This Force Model responds to orders issued by the rule-based agents. Furthermore, the rule-based agents can monitor the conflict situation as the simulation unfolds and issue new orders in response to changes in that situation. In this manner the rule-based agents and Force Model can operate in concert to conduct an automated war game.

The RSAC system permits humans to augment and/or replace any rule-based agent. The RSAC system can therefore serve as a mechanism to conduct manual as well as automated war games.

SLIDE 3: GOALS AND APPROACH

General RSAC goals

- **Capture best features of war gaming and analytic modeling**

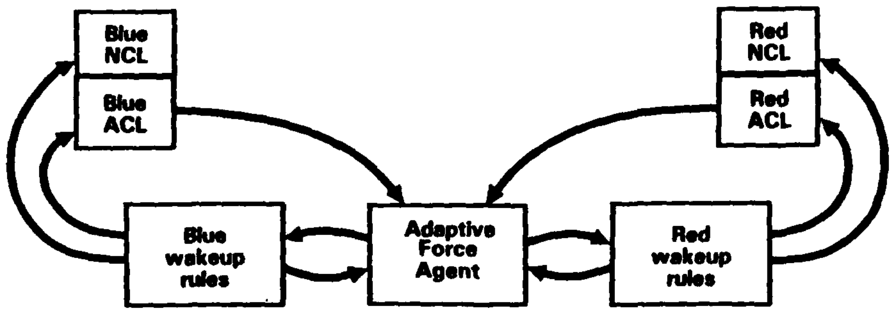
Overall approach

- **Rule-based agents characterize adversary behavior**
- **These agents interact with a coarse-grained Force Model and adapt to changing world situation**
- **Human teams can augment and/or replace the rule-based agents**

The RSAC is attempting to improve strategy analysis by combining the contextually rich discipline of war gaming with the systematic and reproducible precision of analytic modeling and digital simulation. War gaming, by itself, with its contextual richness and realism, does not provide the systematic reproducibility needed to conduct multiscenario analysis. Digital simulation, taken by itself, provides this systematic reproducibility but does not treat the realism found in war gaming. The RSAC system attempts to combine the best of both disciplines.

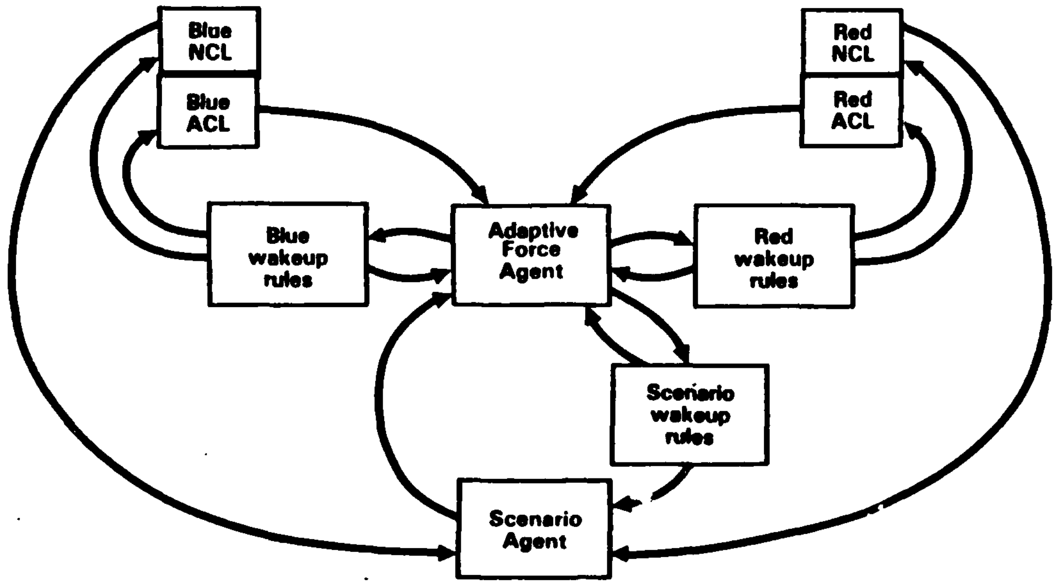
We use the term *multiscenario analysis* rather than *sensitivity analysis* to reflect a subtle difference in the two. *Sensitivity analysis* implies the systematic change of one or more of an analytic model's parameters. *Multiscenario analysis* goes further to also include the systematic change of the setting itself, i.e., usually exogenous factors such as national behavior patterns, which are difficult to capture in the quantitatively precise world of analytic modeling.

SLIDE 12: RSAC SYSTEM INTERFACES MAJOR AGENTS TO FORCE WITH MAJOR AGENT WAKEUP RULES FOR BOTH THE ACL AND NCL



The major point of this slide is that each major agent has both ACL and NCL wakeup rules. ACL wakeup rules detect *expected* (or at least *scheduled*) situations during a plan's implementation, situations at which the plan builders anticipated the need for the ACL to make some operational choices. NCL wakeup rules, on the other hand, detect situations that are *unexpected* (or at least *unscheduled*), i.e., world situations that are no longer consistent with the plan's premises. When a major agent's NCL wakeup rule fires, that agent's NCL will have the opportunity either to change plans or make parametric changes to the current plan.

SLIDE 13: RSAC SYSTEM INTERFACES MAJOR AGENTS TO FORCE WITH MAJOR AGENT WAKEUP RULES FOR BOTH THE ACL AND NCL AND SCENARIO AGENT INTERACTION



This last slide adds the complications implied by an independent *Scenario Agent* that represents independent third-country actions. Just as each major agent has its own set of wakeup rules, so, too, does the Scenario Agent. Those wakeup rules are also tested at each tick of the exercise clock. Scenario Agent can also issue orders to the Force Agent and adapt to the unfolding world situation.

At present Scenario Agent cannot make requests of the major agents. This is consistent with our approach of restricting Scenario Agent to a supportive role to discourage excessive focus on political issues. Later we may allow Scenario Agent more latitude.

Before leaving this series of slides we wish to make three additional points. First, the Force Agent controls the exercise clock. Exercise time is *frozen* when Red, Blue, or Scenario Agent has control. Facilities do exist for the major agents to schedule the issuance of orders at some time in the future, i.e., to impose time delays on the issuance of orders.

Second, while a rule-based agent's wakeup rules are written in ABEL by the agent's rule-writer, the wakeup rules are separate from the agent. From a system architecture perspective the wakeup rules can be thought of as part of the Force Agent.

Third, we wish to underscore the focal role that Force Agent plays in this view of the RSAC system. Not only does the Force Agent keep time, it also manages the precombat and combat interactions among the superpower and third-country military assets and monitors the wakeup rules to determine when the other agents will gain control to adapt to the unfolding world situation.

As other parts of this briefing will show, there are two additional RSAC system perspectives, each with a different central focus. To understand fully how the RSAC system operates, we must understand all three.

SLIDE 14: BRIEFING OUTLINE

- **Brief review**
- **RSAC system interfaces**
- ✓ **Managing interface complexity**
- **System control**
- **Interactive games**

As previous slides indicate, various software agents can interface in a number of complex ways. The next slides describe our approach to managing that interface.

SLIDE 15: MANAGING SYSTEM INTERFACE COMPLEXITIES: THE WORLD SITUATION DATASET

- **No direct inter-Agent communication**
- **All interfaces via World Situation Dataset (WSDS)**
  - **Orders to Force**
  - **Requests to Scenario**
- **WSDS includes all data items each agent needs to perform the exercise**
- **As exercise proceeds, data items take on different values**
- **Data Dictionary controls access to WSDS data items**
  - **Permits AWPs to access WSDS data symbolically**
  - **Enforces access restrictions, e.g., Red cannot examine Blue data**
- **Data Editor provides user "window" on exercise, i.e., on WSDS**

A fundamental feature of our approach is that *we do not permit direct interagent communication*. Rather, all interagent communication is indirect, via what we call the *World Situation Dataset* (WSDS). The WSDS contains all data items relevant to a specific exercise.

Interagent communication takes place in the following ways. The Red and Blue Agents issue orders to Force by effectively changing the *desired* state of specific military assets. Both the actual and desired states of these assets are data items in the WSDS. The Red and Blue Agents make third-country requests in a similar manner.[1] The Scenario Agent, when responding to major agent third-country requests, also issues orders to the Force Agent by altering the WSDS. Force Agent, when conducting its simulation, examines the WSDS, searching for those military assets whose actual and desired states differ. The Force Agent will gradually alter the WSDS to reflect the changed state of military assets.

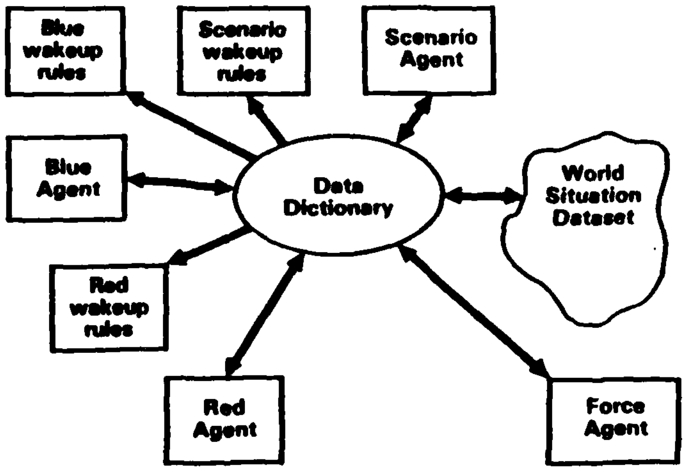
[1]The rule-writer in fact employs primitive functions available in ABEL to make all Force and Scenario Agent WSDS data item changes. He need not worry about the data items directly (in terms of computer storage locations) but can instead refer to them by *name*.

The RSAC system includes a software module called the Data Dictionary to manage access to the WSDS. Through the Data Dictionary the rule-writer can define his data items, impose access restrictions, and provide commentary about each data item's meaning. The Data Dictionary thereby serves not only as an automated means of creating a WSDS and managing access thereto but also as a means for documenting the contents of the WSDS, which may change between RSAC exercises as rule sets change.

The Data Dictionary's access restrictions safeguard the WSDS from unauthorized access. Thus, Red cannot examine Blue's true world situation, but only Red's *perception* of Blue's situation. This perception can differ radically from the *true* world situation to reflect poor intelligence.

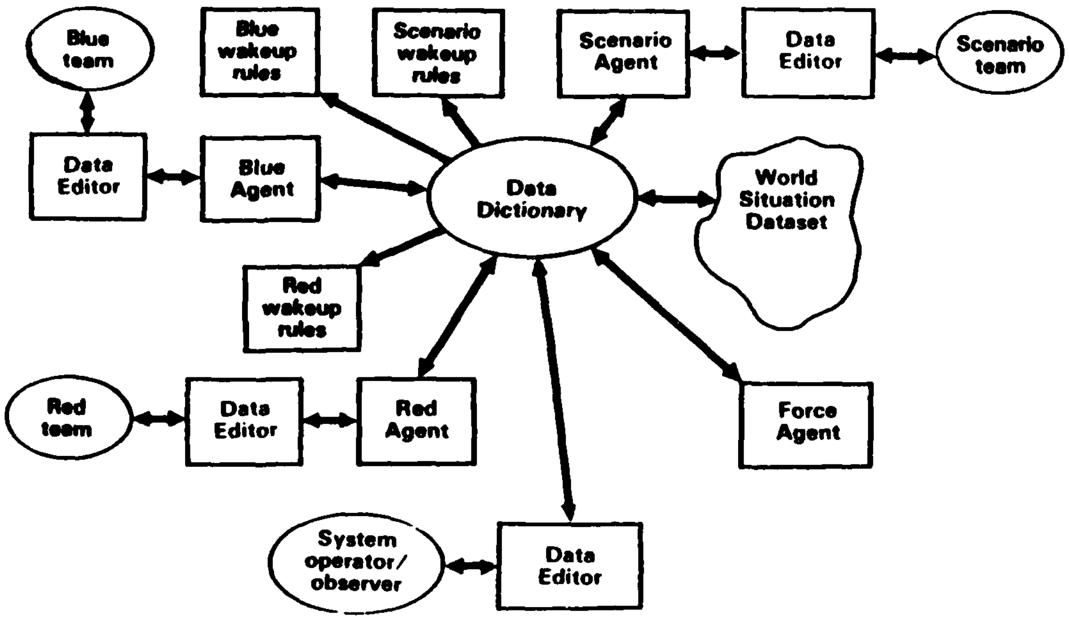
The RSAC system also includes another software module, the Data Editor, which allows human observers/players to interact with the RSAC system through the WSDS. The Data Editor permits the user call standard displays or prepare new ones. These displays can be used passively, to view the unfolding world situation, or actively, with the user (analyst and/or human teams) directly interacting with the system as the exercise progresses.

SLIDE 16: RSAC SYSTEM DATA PATHS



This and the next slide illustrate how the various agents interact with the WSDS via the Data Dictionary. This slide presents the automated exercise situation, and the next slide adds to this the human team situation. Note that this slide distinguishes between the three rule-based agents and their rule-based wakeup rules. Note further that the wakeup rules show only one-way communication paths with the Data Dictionary. They can examine WSDS data items, but they cannot change those data items. Only the agents themselves have that privilege. Finally, note that only the Data Dictionary can directly access the WSDS.

SLIDE 17: RSAC SYSTEM DATA PATHS AND HUMAN INTERACTION



This slide builds on slide 16 by adding human interaction. While the slide shows four distinct human interfaces, one for each rule-based agent and one for the system operators and observers, those interfaces really come in two flavors. The first, for system operators and observers, interfaces directly with the Data Dictionary and gives complete access to the World Situation Dataset, providing operators and observers the detailed status of each agent.

The second type of human interface, which we illustrate as interacting with each rule-based agent rather than directly with the Data Dictionary, represents how human players will interface with the RSAC system. This interface presents the human players with filtered views of the exercise, views that reflect the restricted access those players have to the Data Dictionary, and World Situation Dataset.

When human players replace a Red or Blue automated agent, those players must operate within the context of the NCL. The human players must select from the alternatives normally available at the NCL level, i.e., from the war plans available to the NCL. When making this selection, the human players can ask the RSAC system to forecast the outcome of selecting a specific war plan, i.e., they can ask for *look-aheads*. Slide 19 examines look-aheads in more detail.

SLIDE 18: WHY INTERFACE AGENTS VIA WORLD SITUATION DATASET AND DATA DICTIONARY

- **Well-defined interface**
- **Permits module separability**
  - **Within a single processor**
  - **Among multiple processors**
- **Facilitates postexercise analysis**
- **Permits ad hoc display development (via Data Editor)**
- **Provides ready vehicle for human interaction/intervention**
- **Plan evaluation lookaheads easy to manage**

There are a number of advantages in using a Data Dictionary/WSDS approach. First, it imposes a centralized discipline forcing all development teams to agree precisely on the so-called *primitives* of each interface, i.e., on the syntax and semantics of communication. Second, it promotes a high degree of agent separability, which lays the groundwork for future RSAC system migration from a single processor to multiple distributed processors that could be geographically separated.

Third, the Data Dictionary/WSDS approach facilitates analysis. Because the RSAC system archives the WSDS at all major points in the exercise, it thereby provides an historical record for deliberate, careful post-exercise scrutiny.

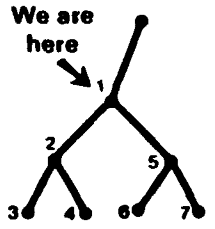
The WSDS is also available during the exercise for such scrutiny, and the Data Editor permits the user to quickly construct ad hoc displays. The system operator can roll an exercise back to an earlier point simply by indicating the WSDS he wishes to reimpose. Human teams can also employ the WSDS, Data Dictionary, and Data Editor to interact with the exercise.

A final advantage arises in managing plan-evaluation look-aheads. The next slide illustrates this.

SLIDE 19: LOOK-AHEADS

At Node 1

1. **Save WSDS (WSDS 1)**
2. **Try path 1-2**
3. **If path 1-2 fails,**
   a. **Restore WSDS 1**
   b. **Try path 1-5**
4. **If path 1-5 fails,**
   a. **Restore WSDS 1**
   b. **Cause NCL wakeup rule to fire**
5. **If path 1-2 succeeds,**
   a. **Restore WSDS 1**
   b. **Implement Path 1-2**
6. **If path 1-5 succeeds,**
   a. **Restore WSDS 1**
   b. **Implement path 1-5**



At Node 2

1. **Save WSDS (WSDS 2)**
2. **Try path 2-3**
3. **If path 2-3 fails,**
   a. **Restore WSDS 2**
   b. **Try path 2-4**
4. **If path 2-4 fails,**
   a. **Restore WSDS 2**
   b. **Note failure at node 2 (path 1-2 fails)**
   c. **Return to node 1**
5. **If path 2-3 succeeds or path 2-4 succeeds,**
   a. **Restore WSDS 2**
   b. **Note success at node 2 (path 1-2 succeeds)**
   c. **Return to node 1**

This slide illustrates rules for performing a look-ahead at a specific plan node. As the slide indicates, we are at node 1, with two branches available. The slide contains illustrative rules for nodes 1 and 2, but similar rules would be found at the slide's other numbered nodes.

At node 1 two branches are available, branch 1-2 and branch 1-5. We prefer branch 1-2 if the look-ahead forecasts a successful outcome. The first step is to save the current WSDS, which the slide labels WSDS 1. This WSDS is set aside for later recall at the look-ahead's conclusion.

Not only does step 1 save the WSDS. It also switches from the real opponent to the major agent's *perception* of the opponent. For example, suppose this is a Red plan. When rule 1 fires, Red's WSDS is set aside, and Red's opponent is changed from the *real* Blue to what we call *Red's Blue*, i.e., Red's perception of Blue. This change has several elements. First the real Blue plan is changed to Red's Blue plan, a plan prepared by Red rule-writers with Red's point of view in mind. Second, Blue's character is changed from the real character to Red's assessment of

Blue's character, i.e., Red's assessment of Blue's approach to military and political conflict. Finally, Red's view of Blue's world situation replaces the real Blue world situation. In other words, in look-ahead mode the outcome is forecast based on the perceived rather than the real opponent.

The second step then instructs the RSAC system to proceed down branch 1-2. At this point the agent *goes to sleep*, awaiting the outcome of the look-ahead. Rules 3-6 will come into play after look-ahead completion. The agent's going to sleep implies that other agents will have an opportunity to execute. Ultimately the Force Agent will gain control and move the look-ahead exercise down branch 1-2.

When the look-ahead exercise arrives at node 2, the node 2 steps will come into play. As with node 1, node 2 has two possible branches, branch 2-3 and branch 2-4. Also as with node 1, node 2's first action will be to save WSDS 2, the WSDS reflecting the state of affairs at node 2. Since we are already in look-ahead mode, we do not have to do anything about the opponent, i.e., we are already using the look-ahead opponent. The next step is therefore to try branch 2-3. Again, the agent then goes to sleep.

Eventually, as the RSAC system moves the agent through the look-ahead, processing will return to node 2. When we return, two possibilities exist. Either branch 2-3 succeeds or branch 2-3 fails. Node 2 rules 3 and 5 cover these two possibilities. If branch 2-3 fails, then rule 3 will fire, causing the RSAC system to restore WSDS 2 and then try branch 2-4. This branch 2-4 attempt will lead to the agent's again going to sleep. If, on the other hand, branch 2-3 succeeds, then rule 5 will fire. This rule also restores WSDS 2. It also sends a message back to node 1 indicating that the look-ahead forecasts success.

The only node 2 rule we have not discussed is rule 4. This rule can fire only if branch 2-4 fails. When such is the case, the rule restores WSDS 2 and returns to node 1 with the message that the branch 1-2 look-ahead has failed. If branch 2-4, on the other hand, succeeds, then rule 5 will fire, returning to node 1 with the success message.

We now return to node 1 processing. After node 1 asks for a branch 1-2 look-ahead, two possibilities exist. The branch 1-2 look-ahead succeeds, or the branch 1-2 look-ahead fails. Node 1 does not care whether the look-ahead took branch 2-3 or branch 2-4, only whether some successful outcome is forecast for branch 1-2.

Since we have only tried a branch 1-2 look-ahead, it is possible for only two of the node 1 rules to fire, namely, rule 3 or rule 5. If rule 3 fires, indicating failure of the branch 1-2 look-ahead, then the plan restores WSDS 1, tries branch 1-5, and goes to sleep. If rule 5 fires, indicating a successful branch 1-2 look-ahead, then the plan restores WSDS 1, changes our opponent back to the real opponent, and resumes the real game down branch 1-2.

The two node 1 rules not as yet described are rules 4 and 6. Rule 6 will fire if branch 1-5 succeeds. If this rule fires, the plan restores WSDS 1, changes the opponent to the real one, and resumes the real game down branch 1-5. Rule 4 will fire if branch 1-5 fails. If this rule fires, then the plan failed in look-ahead on both branches 1-2 and 1-5. When Rule 4 fires the plan therefore restores WSDS 1, changes from the perceived to the real opponent, and causes the NCL to awaken because the plan does not lead to a successful outcome.

We appreciate the complexity of the above description. We would therefore like to leave this slide with some observations. First, the processing at each node does not have to take place at the same time. Indeed, steps 1 and 2 at each node took place before the look-ahead, while rules 3-6 could not fire until after the look-ahead. Second, when asking for a look-ahead, the rule-writer in effect asks the system to shift to the perceived rather than the real opponent.

Third, note that we have distinguished between *steps* and *rules*. Certain plan statements *always* have their effects felt (steps), while others have their effects felt only when certain conditions are satisfied (rules). In our simplified example, steps 1 and 2 had no conditions attached. *Save the WSDS, and try branch 1-2*, i.e., issue orders to Force Agent and/or requests to Scenario Agent. Rules 3-6, on the other hand, require certain conditions to be true for their execution or firing. Those rules all begin with *If*. Fourth, it is

SLIDE 26: SUMMARY

- **Transportable**
  - **UNIX Operating System**
  - **"C" Programming Language**
  - **Potential for distributed processing**
- **Transparent and realistic**
  - **Data Editor provides easy to use, user friendly interface**
  - **ABEL Rule-Based Language**
- **Modular**
  - **Data Dictionary/World Situation Data Set promotes separability**
  - **Three-level Major Agents**
  - **Modular design of Force Agent**
- **Modifiable**
  - **"C" and UNIX promote modifiability**
  - **Data Dictionary and interface standards promote modifiability**
- **Automated and man-machine operation**
  - **Data Dictionary provides an interface compatible with both modes of operation**
  - **Data Editor provides friendly user interface**

Slide 4 listed RSAC system requirements but did not address how the RSAC system has tried to meet those requirements. We now address that issue.

Transportable means the ability to easily move from one computer system to another. The UNIX Operating System and the C Programming Language help to satisfy this requirement. UNIX is available on a wide variety of computer systems today, from large mainframe computers to minicomputers and microprocessors. Furthermore, the number continues to grow. Because UNIX presents a common system *environment* on the growing number of computer systems that support it, the job of transporting the RSAC system to another UNIX system is eased.

Transparent and realistic mean the ability to understand that which is taking place in the RSAC system as well as the ability to provide a realistic setting. With the Data Editor we try to provide a user-friendly if the rules were developed to play the player, which could tend to bind nations to plans beyond the point where that would be reasonable, then the rule-based agent would indeed have to establish context at the beginning of each move. He would have to jump into the middle of things, i.e., enter the plan in the middle rather than the top. The co-routine capability permits this.

place. Rather, before entering in the middle, we must ensure that the software module's previous context (all the module's local storage and registers) is reestablished. Reestablishing this context makes it appear as if the software module never gave up control. The only indication that things have changed comes from the changed state of the World Situation Dataset. In the RSAC system, the three rule-based agents all need this *jump in the middle* capability. The Force Agent does not.

The co-routine capability permits rule development from two different perspectives: One we call *playing the board*, the other we call *playing the player*. Another way of characterizing the two perspectives is *history independent* and *history dependent*. We will use the analogy of a chess game to describe the two perspectives. A chess player can approach a specific move in two ways. He can assume that he has not seen the chess board before, i.e., that he has no knowledge of previous moves. In this instance his decision about the next move can be based only on the current board situation. The player in other words would be playing the board and would have no historical knowledge of how the board got to look the way it does; his play would be history independent.

Alternatively, the chess player can approach a specific move keeping in mind what his opponent has done in the past. The chess player can recollect his opponent's past moves and perhaps get a feel for his opponent's strategy. In other words, at each move the player would view the game in the context of past moves and would not look at the board as if it were a new, fresh problem. This player would be playing the player, and he would use his knowledge of the game's history in deciding on the next move; he would be history dependent.

Both approaches have merit in an automated rule-based system. If the rules were developed to play the board, then on each rule-based agent move, that agent would begin at the top, i.e., would look at the entire situation from a fresh perspective, not needing the history of past moves. Everything the agent needed to make a decision would have to be gleaned from the World Situation Dataset. For some decisions this might prove unrealistic, since nations tend to stick with plans and since history *should* be a major factor in decisions. On the other hand,

SLIDE 25: SOFTWARE ACCOMPLISHMENTS

● **Co-Routines:**	**Permits execution of several "parallel" Agents within one process. Permits "playing the player" as well as "playing the board".**
● **Data dictionary:**	**Permits symbolic access to/manageme nt of RSAC data.**
● **Data Editor:**	**Highly user friendly and user modifiable display interface.**
● **ABEL:**	**Fast, readable rule-based language written in "C".**
● **Goal Directed Simulation:**	**Adaptive technique permits structured but flexible interface among Agents and Force.**

This slide lists some RSAC software accomplishments. Previous discussion addressed all but the first bullet. We will briefly describe the co-routine capability now. Recall that System Monitor serves as the RSAC system's controlling element. Specifically, System Monitor determines which agent next gets control. Once a rule-based agent gains control it retains that control until it puts itself to sleep. When the agent next awakens, System Monitor will cause it to resume execution just after the point at which it went to sleep. For example, a major agent that goes to sleep after issuing mobilization orders to Force Agent will want to be awakened at a point in the plan that deals with postmobilization decisions, i.e., at the plan's next decisionmaking node, which is the point in the plan just after the ABEL statement that puts the agent to sleep.

The co-routine software provides this capability by permitting the RSAC system to enter a plan by *jumping into the middle*. At first glance this may seem a trivial capability. But indeed it is not. Usually, when entering a software module, we enter from the top, in the software module's *main routine*. This routine has the responsibility for managing the execution of that module. But when we enter a software module in the middle, trying to resume where we have previously left off, then we cannot depend on the main routine to make sure that coordination takes

The EnABELER is a software module that translates ABEL rules into C. The DisABELER is a software module that converts a Data Dictionary and World Situation Dataset into the ABEL statements needed to regenerate that World Situation Dataset. This module is useful between exercises when a rule-writer wishes to alter the World Situation Dataset. The rule-writer can alter the ABEL statements directly to conform with his new Data Dictionary. This permits the creation of a new World Situation Dataset consistent with the altered Data Dictionary.

SLIDE 24: COMPUTING MILIEU

- **UNIX Operating System/"C" Programming Language**
- **VAX 11/780 for Development**
  VAX11/750 for Exercises (classified processing)
- **Secure Link from Rand/Washington to Rand/Santa Monica**
- **Software statistics:**

	Lines of code	Language	Memory required	
– Force	6000	C	500K	
– Scenario	5600	ABEL	200K	
– Red*	1200	ABEL	250K	
– Blue*	1200	ABEL	250K	
– Data Editor	9600	C	600K	
– Data Dictionary/WSDS			600K**	
– ABEL support	1200	C	175K	2100K Total
– Enabeler	10,000	C	125K	
– Disabeler	400	C	13K	

***Per plan**
****450K also counted elsewhere**

This and the next slide try to anticipate some of the hardware and software questions you may have.

We have used the UNIX Operating System and the C Programming Language in developing the RSAC system. Not only does UNIX provide a highly productive software development environment, UNIX has also become broadly available on many minicomputer and microcomputer systems.

We are using two computer systems: the Digital Equipment Corporation's VAX11/780 for software development, and the VAX11/750 for classified work and exercises.

The software comprising the run-time RSAC system (ABEL support and items higher on the list) requires a total of about 2100 kilobytes of random access memory, 600 kilobytes of which comprise the World Situation Dataset.

however, operate within the constraints of the available plans and must take actions consistent with the opponent's available plans. For example, a human team wishing to develop a plan employing unconventional warfare activities can do so only if the opponent plans include rules for responding to such activities.

A third type of human interaction involves a human team replacing one or more third countries. As with the previous two examples and indeed all such examples, the third country actions must be consistent with the Red and Blue Agent plans.

SLIDE 23: DEGREES OF INTERACTION

- **System observer/control team**
  - **Critical intelligence**
  - **Perturbations**
  - **Rule changes**
- **Blue or Red NCL**
  - **Specify/alter opponent model**
  - **Evaluate alternative AWPs**
  - **Select/change AWP to be implemented (including operational level parameters)**
- **Scenario Agent**
  - **Replace automated agents for selected third countries**

This slide lists three ways humans can interact with the RSAC system. As system observers or the control team, humans can alter system constraints by making changes to the World Situation Dataset. For example, such changes might involve the insertion of critical intelligence, perturbations, and rule changes. A perturbation can include incorrect intelligence as well as major alterations of an agent's military assets. All of these actions must reflect due consideration for the affected plans. In other words, the system observer/control team must refrain from taking actions inconsistent with plan premises. While the actions can be inconsistent with the plan *currently* being implemented, which will lead to the NCL's awakening, they must at least be consistent with *some* plan. This implies that those making changes must be aware of plan capabilities.

Another type of human interaction involves replacing an agent's automated NCL with a human team. This team will have a number of degrees of freedom, some of them listed on the slide. The team must,

SLIDE 22: BRIEFING OUTLINE

- **Brief review**
- **RSAC system interfaces**
- **Managing interface complexity**
- **System control**
- ✓ **Interactive games**

We now turn to the subject of interactive games.

If humans wish to interrupt the processing, the RSAC system provides them a way to *tap System Monitor on the shoulder*. Specifically, if a human team indicates that it wishes to interact with the system, the team's Data Editor, which acts as that team's window on the exercise, gives the team the ability to alter the value of a WSDS flag which System Monitor and Force Agent both test. We can indeed think of each Data Editor application as having its own wakeup rule. When the flag is altered, then System Monitor passes control to the appropriate Data Editor application.

wishes to awaken. Finding that Blue's wakeup rule fires, System Monitor passes control to Blue Agent, which then issues some requests to the Scenario Agent and orders to the Force Agent.

System Monitor now has control again. It does not poll Red Agent, but rather *begins at the start of the polling sequence*, polling Scenario Agent again. In fact, whenever System Monitor regains control from any of the agents, it begins at the beginning of the Scenario Agent/Blue Agent/Red Agent polling sequence. Since Blue Agent issued requests to Scenario Agent, System Monitor awakens Scenario Agent, which then services the requests.

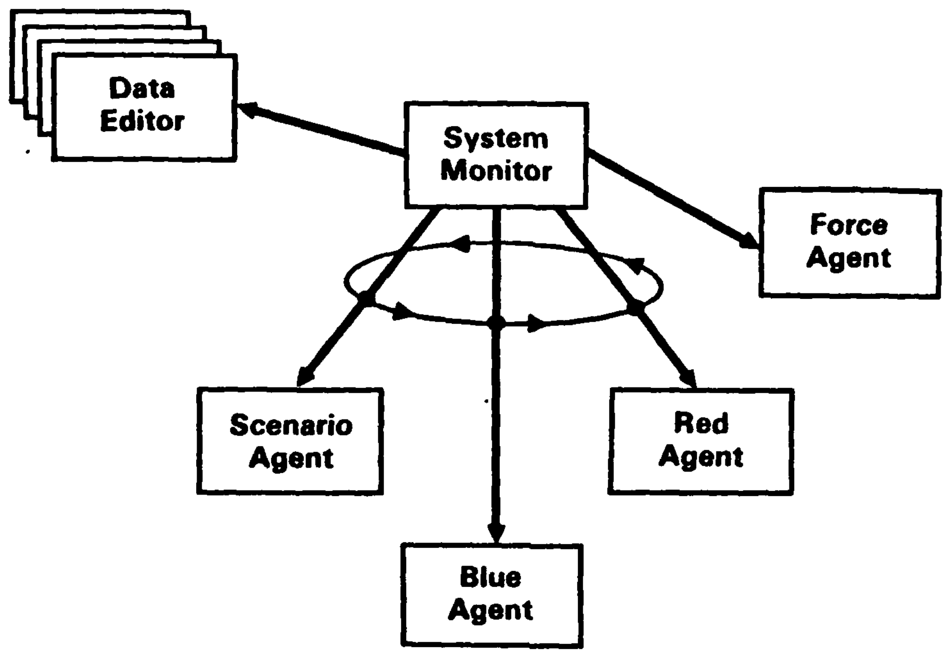
When System Monitor next regains control, it again begins at the beginning of the polling sequence, polling Scenario Agent, Blue Agent, and Red Agent. At this point Red Agent awakens, issuing orders to Force Agent. When Red Agent goes to sleep, System Monitor again regains control. It polls all three agents in the Scenario/Blue/Red order. Since none wishes to awaken, System Monitor passes control back to Force Agent, thereby continuing the exercise.

We start at the beginning of the polling sequence each time to make sure that all rule-based agent actions occur in the *least active to most active* order. We also do not give control back to Force Agent until all rule-based agents pass, i.e., do not wish to awaken. In the automated war games currently contemplated, we see Red as being the most active agent, followed by Blue and ending with Scenario. Hence, the Scenario/Blue/Red polling sequence. However, this polling sequence is not cast in concrete. The RSAC system operator can alter the polling sequence to fit the specific situation.

Only when Force Agent is active does the exercise clock advance. Rule-based agent actions take place in *frozen* time, i.e., the clock does not advance during a rule-based agent's processing.[1]

[1]There are two instances where the exercise clock can move backwards, one under major agent control and one under human team control. Slide 19 described processing when a major agent enters the look-ahead mode, i.e., when that agent wishes to forecast the outcome of an alternative course of action. At the conclusion of a look-ahead, the *look-ahead's* exercise clock can be many days beyond the *real* exercise clock. Restoring the real WSDS restores the real exercise clock. The second instance occurs when a human team requests the rollback of exercise time to a previous decision point.

SLIDE 21: SYSTEM CONTROL



This slide shows System Monitor, the four agents, and several Data Editor applications. To describe System Monitor's function we assume that Force Agent currently has control, conducting the simulation and advancing the exercise clock. Recall that at each tick of the clock, Force Agent examines the active NCL and ACL wakeup rules to determine if any agent needs awakening. We assume that both the Blue and Red Agents now wish to awaken, i.e., Force Agent has found at least one wakeup rule that applies (fires).

Force Agent, upon noting that at least one behavioral agent wishes to awaken, goes to sleep, i.e., relinquishes control to System Monitor. System Monitor then polls the rule-based agent wakeup rules, first polling Scenario Agent's wakeup rules, then Blue's wakeup rules, and finally Red's. Therefore, when System Monitor [illegible]st regains control it asks whether Scenario Agent wishes to awaken. The answer is no, since the Scenario wakeup rule has not fired. Thus System Monitor does not awaken Scenario Agent. System Monitor next asks Blue Agent whether it

SLIDE 20: BRIEFING OUTLINE

- **Brief review**
- **RSAC system interfaces**
- **Managing interface complexity**
- ✓ **System control**
- **Interactive games**

The next slide presents the third and final system perspective, namely, the system control perspective. The first perspective showed the Force Agent as the system's central focus. The second perspective showed the Data Dictionary/World Situation Dataset as the central focus. This third perspective has the System Monitor as its central focus, and System Monitor is responsible for coordinating the execution of the various agents.

possible for some and not other rules to fire during a specific exercise. For example, if branch 1-2 has a successful look-ahead, then all the rules that have to do with trying branch 1-5 will not fire. In the same vein, the rules do not have to fire sequentially. Their firing order depends solely on the set of conditions attached to each. Rule order does not make a difference.

Finally, note that we can have look-aheads within look-aheads. When trying branch 1-2, the plan also performed a look-ahead on branch 2-3 and possibly 2-4. The only restriction that we place on look-aheads within a look-ahead is an administrative one. We do not permit a Red's Blue plan (or a Blue's Red plan for that matter) to ask for a look-ahead. The perceived opponent is very limited in this regard, which we believe reflects the real world situation.

window on the system, a window that the user can adapt to his specific needs. Further, we believe that the ABEL rule-based language also promotes both transparency and realism. The transparency comes from its English-like nature. The realism comes from the rule-writer's ability to specify actions in a natural manner.

Modular means separable. We have attempted to satisfy this requirement in a number of ways. First, each agent is an independent entity with very strict interface requirements. Second, the World Situation Dataset stands between the agents, further enforcing their separability. Third, the three-level structure adopted for the Red and Blue Agents means that parts of them can be altered with minimal impact on the other parts. Finally, because the Force Agent has adopted a goal-directed difference equation approach based on interaction with the World Situation Dataset, that agent's analytic models are separable.

Modifiable means changeable. Several approaches contribute to satisfying this requirement. First, both C and UNIX provide a software environment that promotes sound structured programming practices. Second, the Data Dictionary and World Situation Dataset minimize the degree of *direct* interaction among the software modules, i.e., make it easy for the software developer to design and implement software that does not possess complex data flows and therefore promotes modifiability. Third, the ABEL rule-based language, because it is English-like, helps the plan reader to understand and modify plans.

Automated and man-machine operation means that RSAC must support both automated and human team exercises. We believe that requiring all orders and requests to pass through the World Situation Dataset simplifies both automated and human team operation. Further, we believe that the Data Editor, with its ability to create new windows on an RSAC exercise, also promotes a user-friendly human interface.

To summarize, we believe the system development environment we chose (UNIX and the C Programming Language), the multiagent/multilayer approach to rule-based agent implementation, and the WSDS method of interfacing agents all contribute substantially to meeting the system requirements.